Listing of the Claims:

1. (Previously Presented) A method for enabling events in a COBOL program, the method comprising:

maintaining, in a COBOL program, an index including a process identifier and an event associated with a child process;

initializing, by the COBOL program, the child process;

placing the child process in a wait state when the child process is initialized;

signaling, by the COBOL program, the child process to run using the process identifier and the event associated with the child process.

2. (Original) The method of Claim 1, wherein the COBOL program signals a technical layer using the process identifier and event associated with the child process and further wherein the technical layer signals the child process to run.

3. (Original) The method of Claim 1, wherein the index maintained by the COBOL program maintains a plurality of identifiers and a plurality of events associated with a plurality of child processes.

4. (Original) The method of Claim 1, wherein the child process is placed in the wait state by a technical layer.

5. (Previously Presented) The method of Claim 2, wherein the technical layer is further defined as a COBOL technical layer in communication with the COBOL program.

6. (Previously Presented) The method of Claim 2, wherein the technical layer is further defined as COBOL library including at least one routine callable by the COBOL program.

7. (Previously Presented) The method of Claim 2, wherein the technical layer is integral to the COBOL program.

8. (Previously Presented) The method of Claim 2, wherein the technical layer is enabled by a COBOL compiler.

9. (Previously Presented) The method of Claim 2, wherein the technical layer is integral to a COBOL compiler.

10. (Previously Presented) The method of Claim 2, wherein the technical layer includes a coordination module operable.

11. (Original) The method of Claim 1, wherein the child process registers the process identifier of the child process with a technical layer.

12. (Original) The method of Claim 11, wherein the child process further registers the event associated with the child process with the technical layer.

13. (Original) The method of claim 1, further comprising maintaining a plurality of child processes wherein the process identifiers and events associated with each of the plurality of child processes is maintained in the index of the COBOL program.

14. (Original) The method of 13, further comprising:

providing a COBOL technical layer having a coordination module operable to coordinate signaling the plurality of child processes;

registering, by the plurality of child processes, with the COBOL technical layer;

signaling, by the COBOL program, the COBOL technical layer to run one or more of the plurality of child processes using the process identifiers and events associated with the child processes; and

coordinating, by the coordination module of the COBOL technical layer, the signaling of the child processes.

15. (Original) The method of Claim 14, wherein method further comprises:

creating a system resource by the COBOL program;

designating the system resource to a process identification of the COBOL program;

giving the system resource from the COBOL program to the child process using the process identifier of the child process; and

taking the system resource by the child process from the COBOL program.

16. (Original) The method of Claim 15, further comprising synchronizing such that the COBOL program completes giving the system resource prior to the child process taking the system resource.

17. (Original) The method of Claim 16, wherein the system resource is defined as a socket connection.

18. (Original) The method of Claim 16, wherein the system resource is defined as a pipe connection.

19. (Original) The method of Claim 16, further comprising:

placing the COBOL program in a wait state after giving the system resource to the child process; and

maintaining the COBOL program in the wait state until the child process takes the system resource.

20. (Previously Presented) A system for coordinating processing in COBOL programs, comprising:

a first COBOL program recorded on a computer-readable medium having a first routine for processing;

a second COBOL program recorded on a computer-readable medium having a second routine for processing; and

a module recorded on a computer-readable medium callable by the first and second COBOL programs, the module maintaining a state sharable between the first and second COBOL programs to coordinate the processing of the first and second routines.

21. (Original) The system of Claim 20, wherein the module is further defined as a COBOL library having routines callable from the first and second COBOL programs.

22. (Original) The system of Claim 20, wherein the module is further defined as a COBOL callable routine.

23. (Original) The system of Claim 20, wherein the module is integral to the first and second COBOL programs.

24. (Original) The system of Claim 20, wherein the module is further defined as COBOL compiler enabled.

25. (Original) The system of Claim 20, wherein the module is further defined as a COBOL pre-compiler program used by the first and second COBOL programs.

26. (Original) The system of Claim 20, wherein the first and second routines are further defined as a first and second jobs and wherein the first and second COBOL programs analyze the state maintained by the technical layer to resolve the processing of the first and second jobs.

27. (Original) The system of Claim 20, wherein the module initiates a first state when the first routine is processing such that the second routine postpones processing in response to the first state of the module.

28. (Original) The system of Claim 20, wherein the module is operable to maintain a plurality of states based upon a task of the first and second routines.

29. (Original) The system of Claim 20, wherein the first and second routines are further defined as a first and second threads and wherein the module maintains the state sharable between the first and second COBOL programs to coordinate the processing of the first and second threads.

30. (Original) The system of Claim 29, wherein the first and second threads process in the same address space in a computer system.

31. (Original) The system of Claim 20, wherein the first and second routines are further defined as a first and second jobs and wherein the module maintains the state sharable between the first and second COBOL programs to coordinate the processing of the first and second jobs.

32. (Original) The system of Claim 31, wherein the first and second jobs process in separate address space in a computer system.

33-46. (Canceled)

47. (Original) A method for a COBOL program to use signal handlers, the method comprising:

registering, by a COBOL language program, a signal handler with an operating system, the signal handler associated with an event; and

executing, by the operating system, the signal handler on the event occurs.

48. (Original) The method of Claim 47, wherein the COBOL language program registers with a register of the operating system.

49. (Original) The method of Claim 47, wherein the signal handler executes a corrective process.

50. (Original) The method of Claim 49, wherein the corrective process is closing a file.

51. (Original) The method of Claim 47, wherein the signal handler executes a notification process.

52. (Original) The method of Claim 47, wherein the event is an input/output error.

53. (Original) The method of Claim 47, further comprising:

creating a memory block:

writing an identifier to the memory block related to a system processes being executed; and

reading from the memory block the identifier to determine the system process executed when the event occurred.

54. (Canceled)

55. (Previously Presented) A system for enabling events in COBOL programs, comprising:

a child process recorded on a computer-readable medium that performs a process;

a COBOL program recorded on a computer-readable medium having an index including a process identifier and an event associated with the child process, wherein the COBOL program initiates the child process, puts the child process in a wait state when initiated, and communicates the process identifier at an appropriate time; and

a technical layer recorded on a computer-readable medium having a register including the process identifier and the event associated with the child process, wherein the technical layer signals the child process to run upon receiving the process identifier.

56. (Previously Presented) The system of Claim 55, wherein the COBOL program obtains the process identifier and the event associated with the child process on initializing the child process.

57. (Previously Presented) The system of Claim 55, wherein the child process registers the process identifier and the event with the COBOL program or the technical layer.

58. (Previously Presented) The system of Claim 57, wherein the COBOL program obtains the process identifier and the event associated with the child process from the technical layer subsequent to initializing the child process when the child process registers with the technical layer.

59. (Previously Presented) The system of Claim 55, wherein the technical layer associates the process identifier with the event using the register and then signals the child process to run based on the process identifier.

60. (Previously Presented) The system of Claim 55, wherein the COBOL program communicates the process identifier and the event, and wherein the technical layer signals the child process to run upon receiving the process identifier and the event.

61. (Previously Presented) The system of Claim 55, wherein the COBOL program puts the child process in the wait state through a request, and wherein the technical layer places the child process in the wait state upon receiving the request from the COBOL program.

62. (Previously Presented) The system of Claim 55, further comprising:

a plurality of child processes wherein the process identifiers and events associated with each of the plurality of child processes are maintained in the index of the COBOL program.

63. (Previously Presented) The system of Claim 62, wherein each of the plurality of child processes registers with the technical layer, wherein the COBOL program signals the technical layer to run one or more of the plurality of child processes using the process identifiers and events associated with the one or more of the plurality of child processes, and

wherein the technical layer coordinates signaling the one or more of the plurality of child processes to run.

64. (Previously Presented) The system of Claim 55, wherein the technical layer is further defined as one of a COBOL technical layer in communication with the COBOL program or a COBOL library including at least one routine callable by the COBOL program.

65. (Previously Presented) The system of Claim 55, wherein the technical layer is integral to the COBOL program or is integral to a COBOL compiler.

66. (Previously Presented) The system of Claim 55, wherein the technical layer is enabled by a COBOL compiler.

67. (Previously Presented) The system of Claim 55, wherein the technical layer performs one or more calls to an operating system, wherein the technical layer defines a bit-level mapping of the one or more calls to interface with the operating system.

68. (Previously Presented) The system of Claim 55, wherein the child processes is one of a subtask, a subprogram, or a subroutine of the COBOL program.

69. (Previously Presented) The system of Claim 55, wherein the child process operates as a thread.